PORTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: [✓ The ACM Digital Library] The Guide

thread + Object + synchronization + freelist + lock

SEARCH

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **thread** **Object** **synchronization** **freelist** **lock**

Found **10,249** of **148,786**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

☐ Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Technical papers: concurrency: Assuring and evolving concurrent programs: annotations and policy
Aaron Greenhouse, William L. Scherlis
May 2002 **Proceedings of the 24th International Conference on Software Engineering**

Full text available: pdf(1.38 MB) Additional Information: full citation, abstract, references, citings, index terms

Assuring and evolving concurrent programs requires understanding the concurrency-related design decisions used in their implementation. In Java-style shared-memory programs, these decisions include which state is shared, how access to it is regulated, the roles of threads, and the policy that distinguishes desired concurrency from race conditions. These decisions rarely have purely local manifestations in code.In this paper, we use case studies from production Java code to explore the costs and ...

2 The performance of an object-oriented threads package
John E. Faust, Henry M. Levy
September 1990 **ACM SIGPLAN Notices , Proceedings of the European conference on object-oriented programming on Object-oriented programming systems, languages, and applications**, Volume 25 Issue 10

Full text available: pdf(1.15 MB) Additional Information: full citation, abstract, references, citings, index terms

Presto is an object-oriented threads package for writing parallel programs on a shared-memory multiprocessor. The system adds thread objects and synchronization objects to C++ to allow programmers to create and control parallelism. Presto's object-oriented structure, along with its user-level thread implementation, simplifies customization of thread management primitives to meet application-specific needs. The performance of thread primitives is crucial for parallel programs with ...

3 The performance implications of thread management alternatives for shared-memory multiprocessors
T. E. Anderson, D. D. Lazowska, H. M. Levy
April 1989 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 1989 ACM SIGMETRICS international conference on Measurement and modeling of computer systems**, Volume 17 Issue 1

Full text available: pdf(1.56 MB) Additional Information: full citation, abstract, references, citings, index terms

Threads ("lightweight" processes) have become a common element of new languages and operating systems. This paper examines the performance implications of several data structure and algorithm alternatives for thread management in shared-memory multiprocessors. Both experimental measurements and analytical model projections are presented. For applications with fine-grained parallelism, small differences in thread management are shown to have significant performance imp ...

4 Practitioner reports: Hard real-time: C++ versus RTSJ
Daniel L. Dvorak, William K. Reinholtz
October 2004 **Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications**
Full text available: pdf(192.97 KB) Additional Information: full citation, abstract, references, index terms

In the domain of hard real-time systems, which language is better: C++ or the Real-Time Specification for Java (RTSJ)? Although standard Java provides a more productive programming environment than C++ due to automatic memory management, that benefit does not apply to RTSJ when using NoHeapRealtimeThread and non-heap memory areas. As a result, RTSJ programmers must manage non-heap memory explicitly. Although that's a common practice in real-time applications, it's also a common source of prog ...

Keywords: architecture, concurrency, programming model, real-time

5 An efficient meta-lock for implementing ubiquitous synchronization
Ole Agesen, David Detlefs, Alex Garthwaite, Ross Knippel, Y. S. Ramakrishna, Derek White
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10
Full text available: pdf(2.00 MB) Additional Information: full citation, abstract, references, citings, index terms

Programs written in concurrent object-oriented languages, especially ones that employ thread-safe reusable class libraries, can execute synchronization operations (lock, notify, etc.) at an amazing rate. Unless implemented with utmost care, synchronization can become a performance bottleneck. Furthermore, in languages where every object may have its own monitor, per-object space overhead must be minimized. To address these concerns, we have developed a meta-lock to mediate access to synchro ...

Keywords: concurrent threads, object-oriented language implementation, synchronization

6 An adaptive, region-based allocator for java
Feng Qian, Laurie Hendren
June 2002 **ACM SIGPLAN Notices , Proceedings of the 3rd international symposium on Memory management**, Volume 38 Issue 2 supplement
Full text available: pdf(266.20 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper introduces an adaptive, region-based allocator for Java. The basic idea is to allocate non-escaping objects in local regions, which are allocated and freed in conjunction with their associated stack frames. By releasing memory associated with these stack frames, the burden on the garbage collector is reduced, possibly resulting in fewer collections.The novelty of our approach is that it does not require static escape analysis, programmer annotations, or special type systems. The appro ...

7 Shared memory objects: Bringing practical lock-free synchronization to 64-bit applications
Simon Doherty, Maurice Herlihy, Victor Luchangco, Mark Moir

July 2004 **Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing**
Full text available: pdf(204.03 KB) Additional Information: full citation, abstract, references, index terms

Many lock-free data structures in the literature exploit techniques that are possible only because state-of-the-art 64-bit processors are still running 32-bit operating systems and applications. As software catches up to hardware, "64-bit-clean" lock-free data structures, which cannot use such techniques, are needed.We present several 64-bit-clean lock-free implementations: *load-linked/store-conditional* variables of arbitrary size, a FIFO queue, and a freelist. In addition to being portab ...

Keywords: 64-bit architectures, 64-bit-clean software, compare-and-swap (CAS), freelists, load-linked/store-conditional (LL/SC), lock-free, memory management, multiprocessors, nonblocking synchronization, population-oblivious, queues, space-adaptive

8 Lock reservation: Java locks can mostly do without atomic operations
Kiyokuni Kawachiya, Akira Koseki, Tamiya Onodera
November 2002 **ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 37 Issue 11
Full text available: pdf(246.29 KB) Additional Information: full citation, abstract, references, citings, index terms

Because of the built-in support for multi-threaded programming, Java programs perform many lock operations. Although the overhead has been significantly reduced in the recent virtual machines, One or more atomic operations are required for acquiring and releasing an object's lock even in the fastest cases.This paper presents a novel algorithm called *lock reservation*. It exploits *thread locality* of Java locks, which claims that the locking sequence of a Java lock contains a very lon ...

Keywords: Java, atomic operation, lock, monitor, reservation, synchronization, thread locality

9 Removing unnecessary synchronization in Java
Jeff Bogda, Urs Hölzle
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10
Full text available: pdf(1.45 MB) Additional Information: full citation, abstract, references, citings, index terms

Java programs perform many synchronization operations on data structures. Some of these synchronization are unnecessary; in particular, if an object is reachable only by a single thread, concurrent access is impossible and no synchronization is needed. We describe an interprocedural, flow- and context-insensitive dataflow analysis that finds such situations. A global optimizing transformation then eliminates synchronizations on these objects. For every program in our suite of ten Java bench ...

10 Static conflict analysis for multi-threaded object-oriented programs
Christoph von Praun, Thomas R. Gross
May 2003 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation**, Volume 38 Issue 5
Full text available: pdf(674.11 KB) Additional Information: full citation, abstract, references, citings, index terms

A compiler for multi-threaded object-oriented programs needs information about the sharing